OCR
Oxford Cambridge and RSA

An OCR endorsed teaching and learning tool

OCR A Level Computer Science

H446 – Paper 1

3

Lists and linked lists

Unit 7
Data structures

PG ONLINE

Objectives

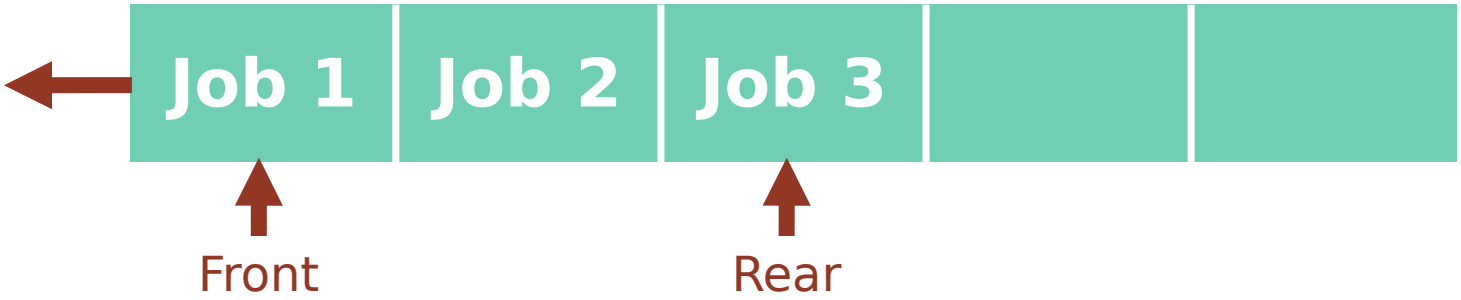
- Explain how a list may be implemented as a static or dynamic data structure
- Describe the linked list data structure
- Show how to create, traverse, add data to and remove data from a linked list

Abstraction

- All programming languages have data types such as real, integer, character and operations which can be performed on them
- In order to deal with complex problems, we need more complex data types that will lead to more efficient problem-solving methods
- This is the idea behind the creation of abstract data types

Abstraction

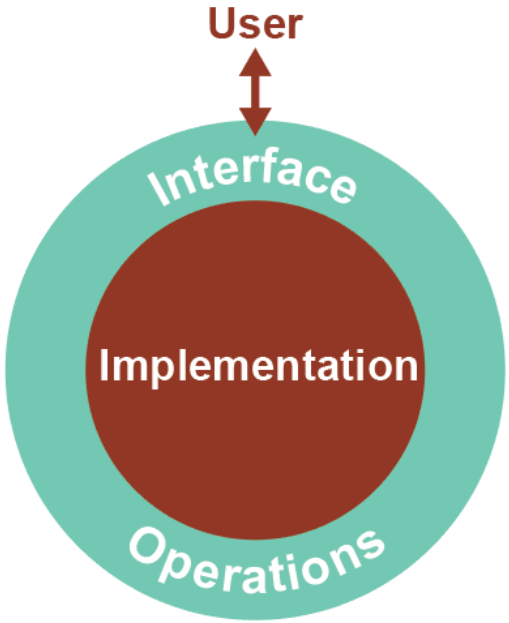
- A queue is an Abstract Data Type (ADT)



- The methods used to implement the queue (e.g. enqueue, dequeue) can be used without knowledge of how they work
- The data (e.g. the pointers) used in implementing a queue are hidden from the user

List – An ADT

- A list is another example of an Abstract Data Type
- Many languages (e.g. Python) have a built-in list data type
- An abstract data type (ADT) allows us to view the data and perform operations that are allowed without regard to how they will be implemented



Dynamic vs static

- In relation to size, what does static mean?
- In relation to size, what does dynamic mean?

Dynamic vs static

- A static data structure cannot change size after it has been created
- A dynamic data structure can grow or shrink
- Programming languages often have an inbuilt dynamic list
- Python, Java, VB.Net, and Delphi all have dynamic list support
  - What controls how many items can be added to a dynamic list?

Implementation of a dynamic list

- The implementation of an inbuilt list ADT is hidden from the user
- A new location is taken from the ‘heap’ - memory locations used for dynamic allocation
- When an item is deleted from a list, the memory location is freed up and returned to the ‘heap’
- A system of pointers keeps the list in the order specified by the user

Applications of lists

- What are some examples of lists in real life and information processing systems?

Applications of lists

- List of students in a class and their marks, component parts of a product, songs, friends, items in a queue, etc.
- "Items in a queue" suggests that we could use the list ADT to implement a queue, with added conditions specified
  - In general, what operations would it be useful to include in a list?

Programming operations

- Can you suggest some operations needed to implement a list? Here are two to get you going

List operation	Operation
isEmpty()	Test for empty list
append(item)	Add a new item to the end of a list

Programming operations

List operation	Operation
isEmpty()	Test for empty list
append(item)	Add a new item to the end of a list
remove(item)	Remove first occurrence of an item from list
count(item)	Return the number of occurrences of item in list
len(item)	Return the number of items in the list
index(item)	Return the position of item
insert(pos,item)	Add a new item at position pos
pop()	Remove and return the last item in the list
pop(pos)	Remove and return the item at position pos

Worksheet 3

- Complete the ‘Random Clothing’ **Task 1** on the worksheet

Sorting a list

- You will cover sorting methods in due course
  - In Python, there is a built-in method for sorting a list, which you can try in interactive mode

```
>>> numbers = [4,78,66,2,99,37]
>>> numbers.sort()
>>> numbers
[2, 4, 37, 66, 78, 99]
```

Implementing a queue as a list

- Using a dynamic data structure such as a list to implement a queue, is there any point in holding items in a circular queue?
- Is it necessary to update the size variable as items are added and removed?
- Do we need a variable maxSize?
- Do we need a function isFull?

Functions to implement a linear queue as a dynamic list

- Write pseudocode to implement the following operations for a queue which can hold a maximum of maxSize items:
  - `enqueue`
  - `dequeue`
  - `isEmpty`
  - `isFull`

Pseudocode

```
procedure enqueue(item)
   if q.isFull() then
         print (“queue
full”)
   else
         q.append(item)
   endif
endprocedure

function isFull()
   return (len(q) ==
maxSize)
endfunction
```

```
procedure dequeue(item)
   if q.isEmpty() then
         print (“queue
empty”)
   else
         q.pop(0)
   endif
endprocedure

function isEmpty()
   return (len(q) == 0)
endfunction
```

Operations on lists

- Merging, sorting, searching and comparing lists are very common operations in computing
  - How could you find how many times numbers in the range
    80 -100 occur in a list of unsorted integers?
  - How could you remove all these numbers from the list?

Worksheet 3

- Try these operations in **Task 2** on the worksheet

Linked lists

- A dynamic abstract data structure which can be implemented as an array and pointers
- Composed of ‘nodes’
- Each node is composed of two parts
  - The data (which may be a complex data structure)
  - A pointer (the index) of the next node
- A start pointer identifies the first node in the list
- A nextfree pointer shows the index of the next free space in the array

Array implementation

- The empty array is intialised as a linked list of free spaces
- start will point to the first element in the list

Index	Data	Pointer
0		1
1		2
2		3
3		4
4		null

start

null

nextfree

0

Adding elements to the list

- We will add the names Nancy, Ava, Dave, Peter to the list
- Start with Nancy

Index	Data	Pointer
0	Nancy	null
1		2
2		3
3		4
4		null

start

0

nextfree

1

Adding elements to the list

- We will add the names Nancy, Ava, Dave, Peter to the list
- Now add Ava

Index	Data	Pointer
0	Nancy	null
1	Ava	0
2		3
3		4
4		null

start

1

nextfree

2

Adding elements to the list

- We will add the names Nancy, Ava, Dave, Peter to the list
- Now add Dave

Index	Data	Pointer
0	Nancy	null
1	Ava	2
2	Dave	0
3		4
4		null

start

1

nextfree

3

Adding elements to the list

- We will add the names Nancy, Ava, Dave, Peter to the list
- Now add Peter. What will be the state of the array and the pointers?

Index	Data	Pointer
0	Nancy	null
1	Ava	2
2	Dave	0
3		4
4		null

start

1

nextfree

3

Adding elements to the list

- We will add the names Nancy, Ava, Dave, Peter to the list
- Now Peter has been added

Index	Data	Pointer
0	Nancy	3
1	Ava	2
2	Dave	0
3	Peter	null
4		null

start

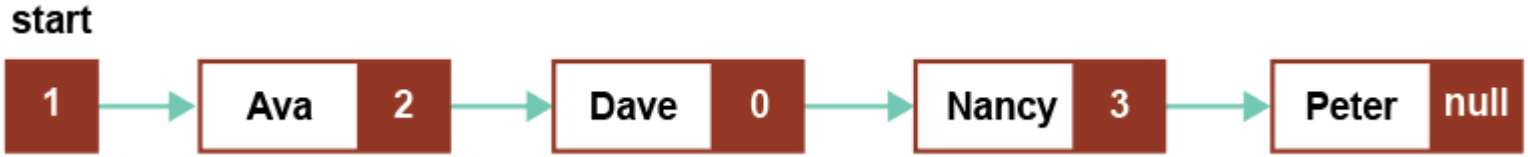
1

nextfree

4

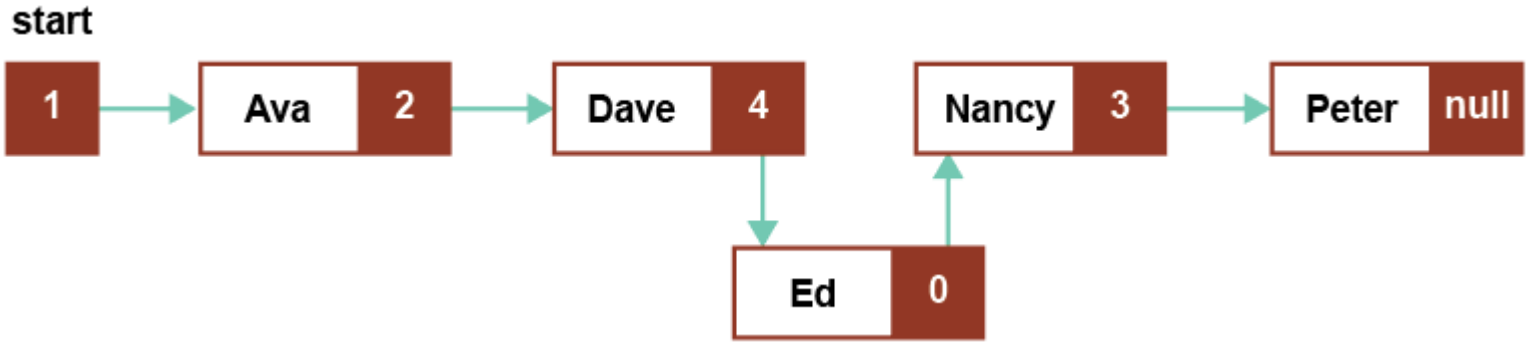
Linked List - diagram

- start points to the head of the list
- Each pointer field holds the index of the next node
- Last node has a null pointer



Adding a new node

- Put the data in the node pointed to by nextfree
- Follow the pointers to find where the new node needs to be linked in
- Adjust the pointers



Deleting a node

- To delete a node, we just need to adjust the pointers
- The deleted node can be linked back in to the list of free nodes by adjusting the pointer in nextfree and the pointer in the deleted node

Peeking ahead

- We can examine the data and the pointer in the current node p and the next one.
  - Suppose p = 1
  - List[p]. Data = Ava
  - List[p].Pointer = 2
  - next = List[p].pointer
  - What is List[next].data?
  - What is List[next].pointer?

Index	Data	Pointer
0	Nancy	3
1	Ava	2
2	Dave	0
3	Peter	null
4		null

start: 1

nextfree: 4

Peeking ahead

- We can examine the data and the pointer in the current node p and the next one.
  - Suppose p = 1
  - List[p]. Data = Ava
  - List[p].Pointer = 2
  - next = List[p].pointer
  - List[next].data = Dave
  - List[next].pointer = 0
- This technique is used in processing a linked list

Index	Data	Pointer
0	Nancy	3
1	Ava	2
2	Dave	0
3	Mike	null
4		null

start: 1

nextfree: 4

Worksheet 3 - Operations

- Complete **Task 3** on the worksheet to develop the algorithms for:
  - Inserting a node
  - Deleting a node

Plenary

- A dynamic data structure such as list is useful for implementing other ADTs such as queues, stacks and trees
  - What is the difference between a static and dynamic data structure?
  - What operations can be performed on a dynamic list?
- You should practise writing and tracing through algorithms for processing a linked list implemented as an array of records

Copyright

Licence agreement

PG ONLINE